On finding approximate supernodes for an efficient ILU(k) factorization

Pascal Hénon, Pierre Ramet and Jean Roman
INRIA, ScAlApplix project - LaBRI, 351 cours de la Libération 33405 Talence, France

Over the past few years, parallel sparse direct solvers have made significant progress [1, 2, 3]. They are now able to solve efficiently real-life three-dimensional problems having in the order of several millions of equations. Nevertheless, the need of a large amount of memory is often a bottleneck in these methods. On the other hand, the iterative methods using a generic preconditioner like ILU(k) (level based factorization) [4] require less memory, but they are often unsatisfactory when the simulation needs a solution with a high precision or when the systems are very ill-conditioned. The incomplete factorization technique usually relies on scalar implementations and thus does not benefit from superscalar effects provided by modern high performance architectures. Futhermore, these methods are difficult to parallelize efficiently. Our goal is to provide a method which exploits the parallel blockwise algorithmic approach used in the framework of high performance sparse direct solvers in order to develop robust parallel incomplete factorization based preconditioners for iterative solvers.

In the direct methods, when the matrix A of the system has a symmetric pattern, a partition $\mathcal{P}$ of the unknowns can be computed from the initial matrix A such that that all columns from unknowns in a same set will have a similar non zero pattern in the factors. The efficiency of a sparse direct solver relies on the fact that one can compute at a low cost the supernode partition before the numerical factorization. This is advantageous in terms of run-time because the block symbolic factorization that computes this dense block structure is a very cheap algorithm compared to the numerical factorization, even if this enables a formulation of the factorization in terms of efficient dense matrix by dense matrix operations (level 3 BLAS routines). In the case of matrices with a symmetric non-zero pattern the block symbolic factorization relies on two important properties:

1. one can find the supernode partition $\mathcal{P}$ from the adjacency graph $G(A)$ [5] with a quasi linear complexity in respect to the number of non-zero terms in the initial matrix A;
2. one can compute the symbolic block structure of L from the quotient graph of A [6] with respect to the supernode partition, that is to say: $Q(G(A), \mathcal{P})^* = Q(G^*(A), \mathcal{P})$.

For a level based incomplete factorization (ILU(k)), these properties do not appear to be true in the general case. Furthermore, the incomplete symbolic ILU(k) factorization has a theorical complexity similar to the one of the numerical factorization. Though this means that it is more difficult to obtain an efficient solver based on a blockwise algorithmic, but some arguments are in favor of this approach. The first one is that an efficient algorithm that leads to a practical implementation has already been proposed [7]. The idea of this algorithm is to search elimination paths of length $k+1$ in $G(A)$ in order to compute $G^k(A)$ which is the adjacency graph of the factor in ILU(k) factorization. The second one is that any set of unknowns in A that have the same row structure and column structure in the lower triangular

part of A can be compressed as a single node in $G(A)$ in order to compute a block symbolic ILU(k) factorization. Indeed the corresponding set of nodes in $G(A)$ will have the same set of neighbors, and consequently the elimination paths of length $k+1$ will be the same for all the unknowns of such a set. In other words, if we consider the partition $\mathcal{P}_0$ constructed by grouping set of unknowns that have the same row and column patterns in A, then we have: $Q(G^k(A), \mathcal{P}_0) = Q(G(A), \mathcal{P}_0)^k$. This is particularly interesting for matrices that come from finite element discretization because in this case a partition $\mathcal{P}_0$ is naturally defined when the unknowns of the matrix are grouped by node in the finite element mesh.

Once the quotient elimination graph $Q(G(A), \mathcal{P}_0)^k$ is computed, the problem is to find a coarser block structure of the incomplete factors. The "exact" supernodes that are exhibited from the incomplete factor non zero pattern are usually very small and thus the resulting dense blocks are not large enough for an efficient use of the BLAS3 routines. A remedy to this problem is to merge supernodes that have nearly the same structure. Thus the principle of this approach is to allow a percentage α of additional non-zeros in the factors; then we propose some heuristic to merge as much supernodes as possible until the additional non-zeros allowed in the factor reach the tolerance α.

This process induces some *extra fill-in* compared to the scalar ILU(k) but the increase of the number of operations is largely compensated by the gain in time due to superscalar effects. The principle of our heuristic to compute the new supernode partition is to iteratively merge supernodes for which non zero patterns are the most similar until we reach a desired extra fill-in tolerance. The cost of our heuristic is really low since it only requires to maintain a sort of the supernodes in respect to their cost to be merged with their father at each step of the process.

On an IBM Power5 SMP node (16 processors with shared memory), this approach permits to solve the $AUDI$ test case ($n = 943695$, $nnzA = 39297771$, PARASOL collection) with a precision of $1e^{-7}$ in about 67 seconds and for a fill ratio of 4 times the number of non zeros in A whereas on the same test case the direct solver takes around 165 seconds and the fill ratio is around 31 times A in this case. The speed-up factor obtained in most test cases was greater than 10 for 16 processors what is pretty good. Another remark that was constated from the tests is that the extra fill-in added by the "supernode coarsener" algorithm allows to decrease the number of iterations in addition to the improvment of the BLAS efficiency.

References

[1] Anshul Gupta. Recent progress in general sparse direct solvers. *Lecture Notes in Computer Science*, 2073:823–840, 2001.

[2] P. Hénon, P. Ramet, and J. Roman. PaStiX: A High-Performance Parallel Direct Solver for Sparse Symmetric Definite Systems. *Parallel Computing*, 28(2):301–321, January 2002.

[3] P. Hénon, P. Ramet, and J. Roman. Efficient algorithms for direct resolution of large sparse system on clusters of SMP nodes. In *SIAM Conference on Applied Linear Algebra, Williamsburg, Virginie, USA*, July 2003.

[4] Y. Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. SIAM, 2003.

[5] Joseph W. H. Liu, Esmond G. Ng, and Barry W. Peyton. On finding supernodes for sparse matrix computations. *SIAM J. Matrix Anal. Appl.*, 14(1):242–252, 1993.

[6] P. Charrier and J. Roman Algorithmique et calculs de complexit pour un solveur de type dissections embotes. *Numerische Mathematik*, 55:463–476, 1989.

[7] D. Hysom and A. Pothen. Level-based Incomplete LU factorization: Graph Model and Algorithms. Tech Report UCRL-JC-150789, Lawrence Livermore National Labs, Nov 2002.